# Novel algorithm to calculate hypervolume indicator of Pareto approximation set

Qing Yang[1] and Shengchao Ding[2,3]

[1] School of Computer Science and Technology, South-Central University for Nationalities, Wuhan, China
[2] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[3] Graduate University of the Chinese Academy of Sciences, Beijing, China
dingshengchao@ict.ac.cn

**Abstract.** Hypervolume indicator is a commonly accepted quality measure for comparing Pareto approximation set generated by multi-objective optimizers. The best known algorithm to calculate it for $n$ points in $d$-dimensional space has a run time of $O(n^{d/2})$ with special data structures. This paper presents a recursive, vertex-splitting algorithm for calculating the hypervolume indicator of a set of $n$ non-comparable points in $d > 2$ dimensions. It splits out multiple child hyper-cuboids which can not be dominated by a splitting reference point. In special, the splitting reference point is carefully chosen to minimize the number of points in the child hyper-cuboids. The complexity analysis shows that the proposed algorithm achieves $O((\frac{d}{2})^n)$ time and $O(dn^2)$ space complexity in the worst case.

## 1 Introduction

Optimization for multiple conflicting objectives results in more than one optimal solutions (known as Pareto-optimal solutions). Although one of these solutions is to be chosen at the end, the recent trend in evolutionary and classical multi-objective optimization studies have focused on approximating the set of Pareto-optimal solutions. However, to assess the quality of Pareto approximation set, special measures are needed [1].

Hypervolume indicator is a commonly accepted quality measure for comparing approximation set generated by multi-objective optimizers. The indicator measures the hypervolume of the dominated portion of the objective space by Pareto approximation set and has received more and more attention in recent years [2, 3, 1, 4].

There have been some studies that discuss the issue of fast hypervolume calculation [5–8]. These algorithms partition the covered space into many cuboid-shaped regions, within which the approach considering the dominated hypervolume as a special case of Klee's measure problem is regarded as the current best one. This approach [8] adopts orthogonal partition tree which requires $O(n^{d/2})$ storage and streaming variant [9]. Conceptual simplification of the implementation are concerned and thus the algorithm achieves an upper bound of

$O(n \log n + n^{d/2})$ for the hypervolume calculation. Ignoring the running time of sorting the points according to the $d$-th dimension, $O(n \log n)$, the running time of this approach is exponential of the dimension of space $d$.

This paper develops novel heuristics for the calculation of hypervolume indicator. Special technologies are applied and the novel approach yields upper bound of $O((\frac{d}{2})^n)$ runtime and consumes $O(dn^2)$ storage. The paper is organized as follows. In the next section, the hypervolume indicator is defined, and some background on its calculation is provided. Then, an algorithm is proposed which uses the so-called vertex-splitting technology to reduce the hypervolume. The complexities of the proposed algorithm are analyzed in Section 4. The last section concludes this paper with an open problem.

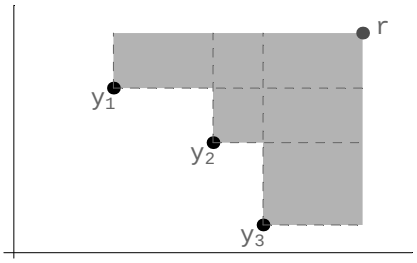
## 2 Background

Without loss of generality, for multi-objective optimization problems, if the $d$ objective functions $f = (f_1, \ldots, f_d)$ are considered with $f_i$ to be minimized, not one optimal solution but a set of good compromise solutions are obtained since that the objectives are commonly conflicting. The compromise solutions are commonly called Pareto approximation solutions and the set of them is called the Pareto approximation set. For a Pareto approximation set $M = \{y_1, y_2, \ldots, y_n\}$ produced in a run of a multi-objective optimizer, where $y_i = (y_{i1}, \ldots, y_{id}) \in M \subset \mathbf{R}^d$, all the solutions are non-comparable following the well-known concept of Pareto dominance. Specially, we say that $y_i$ dominates $y_k$ at the $j$-th dimension if $y_{ij} < y_{kj}$.

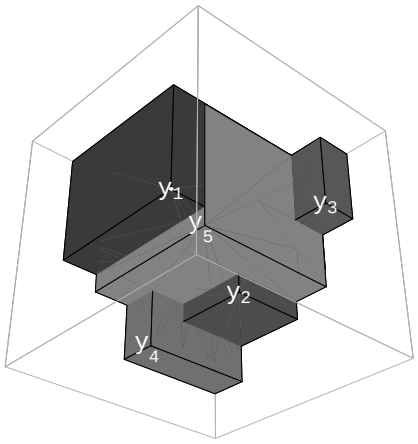
The unary hypervolume indicator of a set $M$ consists of the measure of the region which is simultaneously dominated by $M$ and bounded above by a reference point $r = (r_1, \ldots, r_d) \in \mathbf{R}^d$ such that $r_j \geq \max_{i=1,\ldots,n}\{y_{ij}\}$. In the context of hypervolume indicator, we call the solutions in $M$ as the dominative points. As illustrated in Fig. 1(a), the shading region consists of an orthogonal polytope, and may be seen as the union of three axis-aligned hyperrectangles with one common vertex, i.e., the reference point $r$. Another example in three dimensional space is shown in Fig. 1(b), where five dominative points, $y_1 = (1, 2, 3), y_2 = (4, 3, 2), y_3 = (5, 1, 4), y_4 = (3, 5, 1), y_5 = (2, 2, 2.5)$, and the reference point $r = (6, 6, 6)$ are considered. The volume is the union of the volumes of all the cuboids each of which is bounded by a vertex, where the common regions are counted only once. If a point $y_k$ is dominated by another point $y_i$, the cuboid bounded by $y_k$ is completely covered by the cuboid bounded by $y_i$. And thus only the non-dominated points contribute to the hypervolume.

## 3 The proposed algorithm

In other works, e.g. the work of Beume and Rudolph [8], the hyper-cuboid in $d$-dimensional space are partitioned into child hyper-cuboids along the $d$-th dimension and then all these child hypervolumes are gathered together by the inclusion-exclusion principle [10].




(a) A hypervolume indicator in the two-objective case




(b) A hypervolume indicator in the three-objective case. To lay out the cuboids well, the axes are rotated where the reference point is shaded

In this paper, we step in another way. The hyper-cuboid is partitioned into child hyper-cuboids at some splitting reference points and then all the child hypervolumes are gathered directly. More detailed, given a point $y_i \in M$, each of other points in $M$ must dominated $y_i$ at some dimensions for the non-comparable relation. If the parts over $y_i$ are handled, the problem of calculating the hypervolume bounded by $M$ and the reference point is figured out. The additional part partitioned out at the $j$-th dimension is also a $d$-dimensional hyper-cuboid whose vertices are ones beyond $y_i$ at such dimension. Their projections on the hyperplane orthogonal to dimension $j$ are all dominated by $y_i$, and thus are free from consideration. It should be noted that the reference point of child hyper-cuboid is altered to $r' = (r_1, \ldots, y_{ij}, \ldots, r_d)$, namely the $j$-th coordinate is replaced by $y_{ij}$. The other child hyper-cuboids are handled in the similar way. In these processes, the given point is called the splitting reference point.

Obviously, the hyper-cuboids with more dominative points require more run time to calculate the hypervolumes. To reduce the whole run time for calculating all these child hyper-cuboids, the splitting reference point should be carefully selected. The strategy adopted in this paper is described as follows.

(1) Let $k = n - 1$ and choose a point with the least dimensions on which the point dominated by other $k$ points.
(2) If some points tie, update $k$ as $k - 1$ and then within these points, choose a point with the least dimensions on which the point dominated by other $k$ points.
(3) Repeat the similar process until only single point is left or $k = 1$. And if $k = 1$ and several points are left, the first found point is selected.

By the above principle, as an example, not $y_2$ or other points but $y_5$ is chosen as the first splitting reference point for the case shown in Fig. 1(b). Two child cuboids each bounded by one points and another child cuboid bounded by two points are generated by splitting along $y_5$. This is the optimal strategy in such case.

The algorithm to calculate the hypervolume is shown in Algorithm 1. Some major parameters are as follows.

- **int[n][d] order** The orders of all the dominative points at each dimension are represented by a two-dimensional array of integer.
- **int split** The index of the point at which the hyper-cuboid is cut to generate multiple child hyper-cuboids is called *split*.
- **int[n] splitCount** The numbers of $k$ present in the *split*-th row of the array *order* are saved in *splitCount*, where $k = 0, \ldots, n-1$.
- **int[n] coveredCount** The numbers of $k$ present in the current checked row of the array *order* are save in *coveredCount*, where $k = 0, \ldots, n-1$.

Moreover, some conventions are explained as follows.

- The subscript of $y_{ij}$ begins with 1 while the index of array begins with 0. Thus $y_{ij}$ is same as $y[i-1][j-1]$.
- Assume $a$ and $b$ are two arrays and $n$ is an element. $a[] \Leftarrow n$ means setting each element of $a$ as $n$, while $a[] \Leftarrow b[]$ means copying all the elements of $b$ to $a$ pairwise.
- Assume $S$ is a set and $x$ is an element. $S \Leftarrow S + \{x\}$ means appending a copy of $x$ to $S$.

The inputs of the algorithm are a set of non-dominated (dominative) points and a reference point, thus the hyper-cuboids are represented implicitly.

In fact, when the hyper-cuboid is cut into two child hyper-cuboids, there may be some points dominated by the splitting reference point in the bigger cuboid, and thus such points could be removed from the points set $H$. In the proposed algorithm, it does not matter whether those points are removed or not.

## 4 Complexity Analysis

Before discussing the time-space complexity of the proposed algorithm, some properties are presented firstly.

**Lemma 1.** *Let $\delta_{ij}$ be the number of points dominating $y_i$ at the $j$-th dimension. Then*

*(1) For $d \geq 2$ and each $i \in \{1, \ldots, n\}$, $\sum_{j=1}^{d} \delta_{ij} \geq n-1$.*
*(2) For $d \geq 2$ and each $j \in \{1, \ldots, d\}$, $\sum_{i=1}^{n} \delta_{ij} \leq \frac{n}{2}(n-1)$.*
*(3) For $d = 2$ and each $i \in \{1, \ldots, n\}$, $\sum_{j=1}^{d} \delta_{ij} = n-1$.*
*(4) $\sum_{i=1}^{n} \sum_{j=1}^{d} \delta_{ij} \leq \frac{dn}{2}(n-1)$.*
*(5) For $d \geq 2$ and each $i \in \{1, \ldots, n\}$, $\sum_{j=1}^{d} \delta_{ij} \leq \frac{d}{2}(n-1)$.*

**Algorithm 1** Calculate Hypervolume, $CalcVolume(H)$

**Input:** The hyper-cuboid $H$ defined by the dominative points $\{y_1, y_2, \ldots, y_n\}$ where $y_i = (y_{i1}, \ldots, y_{id})$, and the reference point $r = (r_1, \ldots, r_d)$, namely $H = \{y_1, \ldots, y_n, r\}$. The initial number $n$ of dominative points can be obtained from the length of $H$ and the dimension $d$ is known too.

**Output:** The hypervolume of $H$, *volume*.

```
1: /* initialization */
2: if n = 1 then
3:    return ∏_{j=1}^{d} |r_j − y_{1j}|;
4: end if
5: volume ⇐ 0;
6: splitCount[] ⇐ n;
7: /* count the numbers of points dominating every point at each dimension */
8: for j = 1 to d do
9:    sort y_{1j}, ..., y_{nj};
10:   for all i such that 1 ≤ i ≤ n do
11:      order[i − 1][j − 1] ⇐ number of points dominating y_{ij} strictly;
12:   end for
13: end for
14: /* estimate split based on the statistical results of order */
15: for i = 1 to n do
16:   coveredCount[] ⇐ 0;
17:   for j = 1 to d do
18:      coveredCount[order[i − 1, j − 1]]++;
19:   end for
20:   for k = n − 1 downto 0 do
21:      if coveredCount[k] < splitCount[k] then
22:         split ⇐ i;
23:         splitCount[] ⇐ coveredCount[];
24:         break;
25:      end if
26:   end for
27: end for
28: /* cut H at each dimension through the point indexed by split */
29: for j = 1 to d do
30:   if order[split − 1][j − 1] > 0 then
31:      H2 ⇐ {};
32:      for all y_i in H\{y_{split}, r} do
33:         if y_{ij} is dominated strictly by y_{split,j} then
34:            H2 ⇐ H2 + {y_i};
35:            y_{ij} ⇐ y_{split,j};
36:         end if
37:         /* Here y_i can be removed from H if y_i is dominated strictly by y_{split} */
38:      end for
39:      r2 ⇐ r;
40:      r2[j − 1] ⇐ y_{split,j};
41:      H2 ⇐ H2 + {r2};
42:      volume ⇐ volume + CalcVolume(H2);
43:   end if
44: end for
45: volume ⇐ volume + ∏_{j=1}^{d} |r_j − y_{split,j}|;
46: return volume;
```

*Proof.* It is clear that (2) $\Rightarrow$ (4) $\Rightarrow$ (5). The follows show (1), (2) and (3).

(1) (By Contradiction.) Assume to the contrary there is some $i \in \{1, \ldots, n\}$, $\sum_{j=1}^{d} \delta_{ij} < n-1$. If this is the case, there are at least one $y_k$ where $k \neq i$ such that each $y_{ij}$ dominates $y_{kj}$ for all $j \in \{1, \ldots, d\}$. It follows that $y_i$ dominates $y_k$, which contradicts our assumption that all the elements in $\{y_1, \ldots, y_n\}$ are non-comparable.

(2) Given $j$, sort all $y_{ij}$ where $i = 1, \ldots, n$ and label each $y_{ij}$ a sequence number $I(i)$ which ranges from 0 to $n-1$. Thus $\sum_{i=1}^{n} I(i) = \frac{n}{2}(n-1)$. There are two cases to consider. Firstly, if all $y_{ij}$ are different each other, then $\delta_{ij} = I(i)$. It follows that $\sum_{i=1}^{n} \delta_{ij} = \frac{n}{2}(n-1)$. Secondly, if there are same elements within $\{y_{1j}, \ldots, y_{nj}\}$, without loss of generality, suppose $y_{ij} = y_{kj}$ and $I(k) = I(i) + 1$. Then $\delta_{ij} = \delta_{kj} = I(i) < I(k)$, it follows that $\sum_{i=1}^{n} \delta_{ij} < \frac{n}{2}(n-1)$. This completes the proof.

(3) (By contradiction.) For any $y_i$, $\sum_{j=1}^{d} \delta_{ij} < n-1$ is excluded by (1) of this lemma. Thus $\sum_{j=1}^{d} \delta_{ij} > n-1$ for some $y_i$ is considered. If this is the case, we obtain $\sum_{i=1}^{n} \sum_{j=1}^{d} \delta_{ij} > n(n-1)$, contradicting (2) of this lemma, which implies $\sum_{i=1}^{n} \sum_{j=1}^{2} \delta_{ij} = \sum_{j=1}^{2} \sum_{i=1}^{n} \delta_{ij} \leq n(n-1)$, namely $\sum_{i=1}^{n} \sum_{j=1}^{d} \delta_{ij} \leq n(n-1)$.

**Lemma 2.** *Let $\omega_i(k)$ be the amount of $k$ in all $\delta_{ij}$ where $j = 1, \ldots, d$, namely $\omega_i(k) = |\{j : \delta_{ij} = k, j = 1, \ldots, d\}|$. Then*

*(1) $0 \leq \omega_i(k) \leq d$ for any $i$ and $k$;*
*(2) $\sum_{i=1}^{n} \omega_i(k) \leq d$ for any $k$;*
*(3) $\sum_{k=0}^{n-1} k\omega_i(k) \leq \frac{d}{2}(n-1)$ for any $i$.*

*Proof.* By the definition of $\omega_i(k)$, it is clear that all statements follows Lemma 1.

**Lemma 3.** *Let $f(n,d)$ be the runtime of Algorithm 1 to compute a hypervolume with $n$ dominative points in a $d$-dimensional space. Then*

*(1) $f(n,d) + f(m,d) > f(n-1,d) + f(m+1,d)$ where $n - m > 1$;*
*(2) $f(n,d) > f(m,d) + f(n-m,d)$ where $n > m$;*
*(3) $f(n,d)$ is minimal when $\sum_{j=1}^{d} \delta_{ij} = n-1$ and $|\delta_{ij} - \delta_{ik}| \leq 1$ for any $j$ and $k$;*
*(4) $f(n,d)$ is maximal when $\sum_{j=1}^{d} \delta_{ij} = \frac{d}{2}(n-1)$ for any $i$ and $\omega_i(k) = \frac{d}{n}$ for any $i$ and each $k = 0, \ldots, n-1$.*

*Proof.* (1) and (2) are clear.

(3) By the process of Algorithm 1, given some $i$,

$$f(n,d) = dn \log n + \sum_{j=1}^{d} f(\delta_{ij}, d) \tag{1}$$

By (1) of Lemma 1, $\sum_{j=1}^{d} \delta_{ij} \geq n-1$. It is clear that for a given $i$, it is necessary that $\sum_{j=1}^{d} \delta_{ij} = n-1$ to minimize $f(n,d)$. In addition, all the $\delta_{ij}$

must share alike, i.e. $|\delta_{ij} - \delta_{ik}| \leq 1$ for any $j$ and $k$. If this is not the truth, suppose $\delta_{ij} - \delta_{ik} > 1$. Thus by (1) of this lemma,

$$f(\delta_{ij}, d) + f(\delta_{ik}, d) > f(\delta_{ij} - 1, d) + f(\delta_{ik} + 1, d) \tag{2}$$

Let $\delta_{ij'} = \delta_{ij} - 1$ and $\delta_{ik'} = \delta_{ik} + 1$. $\delta_{ij'}$ and $\delta_{ik'}$ can be modified in the similar way until $|\delta_{ij} - \delta_{ik}| \leq 1$. This completes the proof.

(4) By (5) of Lemma 1, $\sum_{j=1}^{d} \delta_{ij} \leq \frac{d}{2}(n-1)$. It is clear that for a given $i$, it is necessary that $\sum_{j=1}^{d} \delta_{ij} = \frac{d}{2}(n-1)$ to maximize $f(n, d)$.
Hence Eqn. (1) is written as follows,

$$f(n, d) = dn \log n + \sum_{k=1}^{n-1} \omega(k) f(k, d) \tag{3}$$

Suppose $y_i$ is the splitting reference point chosen by Algorithm 1, $\omega_i(n-1) \leq \frac{d}{n}$, or else contradicting $\sum_{i=1}^{n} \omega_i(n-1) \leq d$. To maximize $f(n, d)$ in Eqn. (3), let $\omega_i(n-1) = \frac{d}{n}$. Similarly, we get $\omega_i(n-2) = \frac{d}{n}$, ..., $\omega_i(1) = \frac{d}{n}$, and so on. It is exactly $\sum_{k=0}^{n-1} k\omega_i(k) = \frac{d}{2}(n-1)$. This completes the proof.

### 4.1 Bounds of runtime at special cases

First of all, it is clear that $f(1, d) = d$. By (3) of Lemma 3, the algorithm performs best when each $\delta_{ij}$ shares alike for the chosen $i$. If $d \geq n-1$, $\delta_{ij} \leq 1$ for any $j$. Thus

$$f(n, d) = dn \log n + (n-1) f(1, d) \tag{4}$$

which implies $f(n, d) = \Omega(dn \log n)$. If $d < n-1$, $\delta_{ij} > 1$ for any $j$. In the rough, we get

$$f(n, d) = dn \log n + d \cdot f(\frac{n-1}{d}, d) < dn \log n + d \cdot f(\frac{n}{d}, d) \tag{5}$$

It can be obtained from Eqn. (5) that $f(n, d) = \Theta(dn \log n \log_d n)$ even when $f(\frac{n-1}{d}, d)$ is relaxed to $f(\frac{n}{d}, d)$.

Fredman and Weide [11] have shown that Klee's measure problem has a lower bound of $\Omega(n \log n)$ for arbitrary $d \geq 1$. Just as Beume and Rudolph [8] have mentioned, although it is unknown what the lower bound for calculating the hypervolume is, it is definitely not harder than solving KMP because it is a special case of KMP. Therefore, there is a gap between the lower bound of the proposed algorithm and the actual lower bound of calculating the hypervolume.

In the average cases, suppose that for the given splitting reference point $y_i$, $\sum_{j=1}^{d} \delta_{ij} = \frac{d}{2}(n-1)$. Meanwhile, each $\delta_{ij}$ shares alike, i.e. $\delta_{ij} = \frac{n-1}{2}$. Thus,

$$f(n, d) = dn \log n + d \cdot f(\frac{n-1}{2}, d) < dn \log n + d \cdot f(\frac{n}{2}, d) \tag{6}$$

which implies the runtime of the proposed algorithm is $\Theta(dn^{\log d})$ at the given cases.

### 4.2 Upper bound of runtime

By (2) of Lemma 3, $f(n-1) > f(n-1-k) + f(k)$ for any $k = 1, \ldots, \frac{n-2}{2}$. And by (4) of Lemma 3, at the worst cases, we have

$$\begin{aligned} f(n,d) &= dn\log n + \tfrac{d}{n}\left(f(n-1) + f(n-2) + \ldots + f(2) + f(1)\right) \\ &< dn\log n + \tfrac{d}{n}(1 + \tfrac{n-2}{2})f(n-1) \\ &< dn\log n + \tfrac{d}{2}f(n-1) \end{aligned} \tag{7}$$

which implies that the proposed algorithm for computing the hypervolume bounded by $n$ points and a reference point in $d$-dimensional space has a runtime of $O((\frac{d}{2})^n)$ at the worst cases.

### 4.3 Space complexity

Let $g(n,d)$ be the used storage by Algorithm 1. In the proposed algorithm, every child hypervolume is calculated one by one. Since the storage can be reused after the former computation has been completed, $g(n,d)$ is only related to the maximum usage of all the computations of child hypervolumes. Hence,

$$g(n,d) = dn + \max_{i\in\{1,\ldots,n\}, j\in\{1,\ldots,d\}} \{g(\delta_{ij}, d) : 0 \le \delta_{ij} \le n-1\} \tag{8}$$

Thus the upper bound of space is as follows.

$$g(n,d) = dn + g(n-1,d) \tag{9}$$

where $g(1) = d$. It is easy to obtain an $O(dn^2)$ space upper bound for the proposed algorithm.

Combining the above analyses together, we obtain the time-space complexity of the proposed algorithm.

**Theorem 1.** *The hypervolume of a hyper-cuboid bounded by $n$ non-comparable points and a reference point in $d$-dimensional space can be computed in time $O((\frac{d}{2})^n)$ using $O(dn^2)$ storage.*

## 5 Conclusions

A fast algorithm to calculate the hypervolume indicator of Pareto approximation set is proposed. In the novel algorithm, the hyper-cuboid bounded by non-comparable points and the reference point is partitioned into many child hyper-cuboids along the carefully chosen splitting reference point at each dimension. The proposed approach is very different to the technique used in other works where the whole $d$-dimensional volume is calculated by computing the $(d-1)$-dimensional volume along the dimension $d$. Such difference results in very different time bounds, namely $O((\frac{d}{2})^n)$ for our work and $O(n^{\frac{d}{2}})$ for the best previous result. Neither kind of technique can exceed the other completely and each has his strong point. Additionally, the amount of storage used by our algorithm is

only $O(dn^2)$ even no special technique is developed to reduce the space complexity.

As the context has mentioned, it is very important to choose appropriate splitting reference point for our algorithm. Well selected point can reduce number of points in separated parts and thus cut down the whole runtime. We do not know whether the strategy adopted in this paper is optimal or near optimal. Further investigations should be worked on.

## References


1. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. IEEE Transactions on Evolutionary Computation **7**(2) (2003) 117–132
2. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms — A Comparative Study. In Eiben, A.E., ed.: Parallel Problem Solving from Nature V, Amsterdam, Springer-Verlag (1998) 292–301
3. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation **3**(4) (1999) 257–271
4. Zitzler, E., Brockhoff, D., Thiele, L.: The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In: Proceedings of the 4th International Conference on Evolutionary Multi-Criterion Optimization (EMO 2007). Volume 4403., Springer-Verlag (2007) 862–876
5. While, L., Bradstreet, L., Barone, L., Hingston, P.: Heuristics for optimising the calculation of hypervolume for multi-objective optimisation problems. In: The 2005 IEEE Congress on Evolutionary Computation. Volume 3. (2005) 2225–2232
6. While, L., Hingston, P., Barone, L., Huband, S.: A faster algorithm for calculating hypervolume. IEEE Transactions on Evolutionary Computation **10**(1) (2006) 29–38
7. Fonseca, C.M., Paquete, L., nez, M.L.I.: An improved dimension-sweep algorithm for the hypervolume indicator. In: IEEE Congress on Evolutionary Computation (CEC 2006). (2006) 1157–1163
8. Beume, N., Rudolph, G.: Faster s-metric calculation by considering dominated hypervolume as klee's measure problem. In Kovalerchuk, B., ed.: Proceedings of the Second IASTED Conference on Computational Intelligence, Anaheim, ACTA Press (2006) 231–236
9. Edelsbrunner, H., Overmars, M.H.: Batched dynamic solutions to decomposable searching problems. Journal of Algorithms **6**(4) (1985) 515–542
10. Overmars, M.H., Yap, C.K.: New upper bounds in klee's measure problem. SIAM Journal on Computing **20**(6) (1991) 1034–1045
11. Fredman, M.L., Weide, B.: The complexity of computing the measure of $\bigcup [a_i, b_i]$. Communications of ACM **21** (1978) 540–544